分布式对象存储在采编融媒体平台的探索和应用

高 文 任柏青 魏海涛 周晨赓
（新华通讯社通信技术局，北京 100803）

摘要：【目的】融媒体的飞速发展，极大地推动了多媒体内容的创作和分享，同时随着 4K、5G 和 AI 技术的成熟发展，新华社用户上传和入库的稿件中照片、音乐、视频等文件与日俱增，加剧了数据量的爆发，不断占用着存储资源。【方法】针对这类非结构化数据的存储和管理，传统的块存储（SAN）和文件存储（NAS）由于其本身技术和架构的限制略显捉襟见肘，对象存储应运而生。基于此，文章主要介绍了数据存储的发展过程，以及分布式对象存储的特点及优势，结合新华社采编融媒体平台业务探索应用可行性。【结果】【结论】通过搭建 MinIO 对象存储环境，并改造采编融媒平台文件服务接口，可实现对象存储的平滑切换，验证对象存储在采编融媒平台应用的可行性。

关键词：分布式；对象存储；融媒体；MinIO；采编 中图分类号：TP311 文献标识码：A
文章编号：1671-0134（2023）06-104-05 DOI：10.19483/j.cnki.11-4653/n.2023.06.022
本文著录格式：高文，任柏青，魏海涛，周晨赓．分布式对象存储在采编融媒体平台的探索和应用 [J]. 中国传媒科技，2023（06）：104-107，158.

导语

近年来，进入数字经济时代，随着海量应用的兴起，数据呈现爆炸式增长，数据类型也更加多样复杂。据国际数据公司（IDC）预测，全球数据量预计 2025 年将达到 175ZB。此外，数据的类型也在发生变化，随着技术的各种深入应用，视频、音频、影像等各种非结构化数据的增长更为迅速。

在媒体深度融合的背景下，传统媒体遇到诸多挑战，传统媒体也在新技术的驱动下迎来了升级。作为舆论引导主力军、主阵地、主渠道，新华社在媒体融合发展取得重大进步和显著成效，推出一系列融媒体报道产品。根据新华社每日电讯报道，2021 年全国两会期间新华社国内部“主力军全面挺进主战场”，全链条策划、全媒化采集，同时创新思维发挥长项，拼创意、抢阵地，国内部融媒产品产量“逆转”传统通稿，“破圈突围”，新媒体和通稿发稿比例由 2020 年的 1 : 1 变成了超过 2 : 1。[1-2]

在传统媒体到新媒体的转型过程中，文件大、数量多的非结构化媒体数据存储是一个严峻考验。对媒体来说，数据和内容始终是核心，在新闻素材的采集、编辑、加工、管理、分发，到最终三审一校的成稿等诸多环节中，需要考虑到如何在多用户、多系统间共享，如何支持多业务流程，最终满足融媒体产品制作、流转、审核、备份归档、数据分析、AI 等多方面的需求。

面对着数据爆炸式增长和非结构化数据占比显著增加这两大趋势，传统的存储系统，诸如 DAS、SAN 和 NAS 等集中式存储，在容量和性能等方面因为本身技术和架构的限制，无法进行有效应对。人们需要一种全新架构的存储系统，这种存储系统需具备极高的可扩展性，能够满足人们对存储容量 TB 到 EB 规模的扩展的需求。于是，对象存储应运而生。

1. 数据存储发展

随着信息技术和互联网的发展，存储系统也在不断进化，系统功能越来越丰富，存储容量不断提升，性能也越来越强。通俗地讲，按照存储功能和使用方式的不同，可以分为以下 3 种类型。

1.1 块存储

块存储，通过控制器将多块磁盘重新组成一个或者多个逻辑磁盘，作为块存储设备提供给主机使用，在主机上显示的就是一块或多块硬盘，通常需要主机操作系统对硬盘进行分区和格式化文件系统后才能使用。主机通过数据线直连的块存储，称为 DAS 存储（Direct-Attached Storage，直连式存储），通过光纤通道网络或者 IP 网络访问的块存储，称为 SAN 存储（Storage Area Network，存储区域网络）。块存储的扩展性和灵活性较差。

1.2 文件存储

文件存储，是指在块存储的基础上增加了文件系

统，通过网络远程的方式将文件系统挂载到主机使用，在主机上显示的就是一个文件系统分区，可以直接使用。应用程序无需关心不同文件存储的差别，和访问本地文件系统一样，通过调用操作系统 POSIX 标准文件系统接口来操作文件存储中的目录和文件。众所周知，文件系统中有目录和文件，采用层级树形结构进行文件管理，便于文件查找。用户可以通过操作系统中的应用程序对文件进行打开、修改等操作，如打开 Word、编辑 Word。文件存储具有丰富多样的功能，很适合用来存储文档、图片、视频等各种非结构化文件。

1.3 对象存储

对象存储，最初是公有云厂商提供的云存储产品，通过 Web 接口提供 Key-Value 形式的分布式对象存储服务。可以把任何数据转换成字节流赋值给 Value，并提供一个字符串类型的 Key，一起提交给对象存储保存。在读取时，只需要提供 Key，就可以读取对应的 Value。它更像一个 Web 应用，而不像传统存储。人们经常将其用来存储图片、视频等文件，用于解决海量的互联网文件存储和用户访问。通常，使用时会模仿本地文件系统的使用习惯，将文件路径（含目录路径）字符串作为 Key，将文件数据转换为字节流之后作为 Value，一起提交给对象存储保存，间接实现存储文件的目的。更特别的是，可以按照对象存储的规则，将 Key 放到一个特定 URL 网址里，在浏览器里输入后就可以直接访问，这对图片或者视频而言，是非常便利的。因此，对象存储最主流的使用场景，就是存储网站、移动 App 等互联网/移动互联网应用的静态内容（视频、图片、文件、软件安装包等）。

相较于传统存储，对象存储具备很多新的特性，笔者认为以下几种较为突破：

（1）版本控制功能。对象的每一次更改，都会生成一个新版本，用户可以获取任意一个历史版本，同时提供对象锁、WORM（一次写入多次读取）等功能。

（2）更细粒度的安全控制和审计。在身份鉴权的基础上，具备对象级别的访问控制能力，每一个对象都可以单独设置用户权限和读写权限。另外，由于访问接口为 REST API，更容易获取和分析用户访问日志。[3]

（3）丰富的存储桶策略。存储桶是存放对象的容器，可以针对存储桶设置一系列策略，作用于所有对象，包括启用版本控制、启用保留时间机制、远程复制等。

2. 对象存储系统方案

2.1 采编业务文件存储特点

采编业务中的文件存储主要类型是图片、音频、视频、文档等多媒体文件，文件数量为 2500 万个左右，文件大小从几 KB 到几 GB 不等，实时写入和读取。

2.2 软件定义存储 MinIO

鉴于采编业务文件存储特点，采编融媒体平台开始探索使用对象存储对图片、音频、视频、文档等多媒体文件进行存储和存取访问。采编融媒体平台对象存储系统采用软件定义存储的方式，基于开源分布式对象存储软件 MinIO，通过在普通 PC 服务器上运行对象存储程序，将多台 PC 服务器上的硬盘组成一个存储资源池，对外提供对象存储服务，来实现对象存储功能，从而替代昂贵的传统集中存储设备。[4]

MinIO 是在 GNU AGPLv3 协议下开源的对象存储软件，提供 Amazon S3 兼容接口，支持所有的 S3 特性，支持本地部署和容器部署，支持存储的单个文件大小最大为 50TB，支持的集群存储容量理论上没有上限。MinIO 部署步骤简单，易上手，核心程序就是一个 go 编译的二进制文件，不依赖特定软件和库，通过环境变量来配置节点。支持多地复制，也就是不同的集群之间，支持数据自动复制，来实现异地容灾备份的目的。按照 MinIO 官网自己的说法，MinIO 是世界上最快的对象存储，在 32 个 NVMe 驱动器节点和 100Gbe 网络上发布的 GET/PUT 结果超过 325GB/秒和 165GB/秒。[5]

MinIO 自带 Web 管理界面和命令行工具 mc。可以在 Web 管理界面上对 MinIO 进行对象管理、存储桶管理、用户管理、数据修复、诊断和日志管理。如图 1 所示。

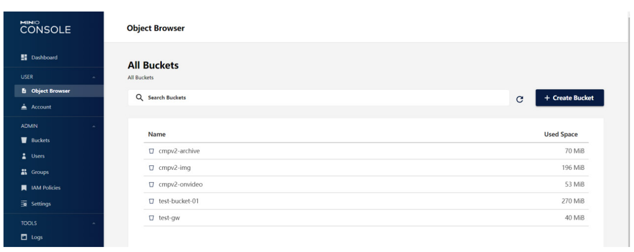


图 1 搭建 MinIO 环境，Web 管理页面

2.3 系统架构和部署

MinIO 提供单节点单硬盘（Single-Node Single-Drive，以下简称 SNSD）、单节点多硬盘（Single-Node Multi-Drive，以下简称 SNMD）、多节点多硬盘（Multi-Node Multi-Drive，以下简称 MNMD）3 种部署方式。SNSD 模式是单机模式，文件会原封不动存储在 MinIO 后端磁盘上，MioIO 只能用于 S3 API 网关，无法使用对象存储的诸多特性。SNMD 和 MNMD 模式

为分布式集群模式，采用纠删码方案来实现冗余和高可用，提供对象级别修复，比 RAID 或者副本方案开销更少。本文采用 MNMD 模式。

MNMD 模式的 MinIO 集群层级元素为集群（Cluster）、服务器池（Server Pool）、纠删集（EC Set）、驱动器（Drive），层级结构如图 2 所示。MinIO 集群由一个或多个服务器池组成，各服务器池独立运行，互不影响，可以通过增加服务器池来进行水平扩容。每个服务器池由多台服务器和磁盘驱动器组成。一个服务器池内，MinIO 会基于服务器数量和磁盘驱动器数量，将所有磁盘驱动器自动分组为不同的纠删集，也就是条带化。一个纠删集中磁盘驱动器数量（条带大小）范围为 2 到 16，一旦初始化之后便固定不变，并且一个服务器池中的所有纠删集都有相同的条带大小。MioIO 接收到对象上传请求后，根据算法选定将对象保存到哪个纠删集，假设该纠删集条带大小为 M（包含 M 个磁盘驱动器），然后实时将对象编码为 M–N 个数据块和 N 个校验块后分别写入该纠删集的各个磁盘驱动器中，每一个磁盘驱动器保存一个数据块或者校验块。一般使用 EC：N 来表示校验块数量，可以容忍 N 个数据块损坏或丢失，当数据块丢失或者损坏的情况下，可以通过校验块来重组数据块。特殊的，最大化校验块配置情况下，也就是 N 等于 M 的一半时，该纠删集可以容忍一半的磁盘驱动器损坏。每个对象只会保存在一个服务器池，不会跨服务器池存储。MinIO 集群提供各服务器池所有对象的统一视图，单个服务器池故障，不会影响整个集群，但单个服务器池离线会影响集群对外服务。

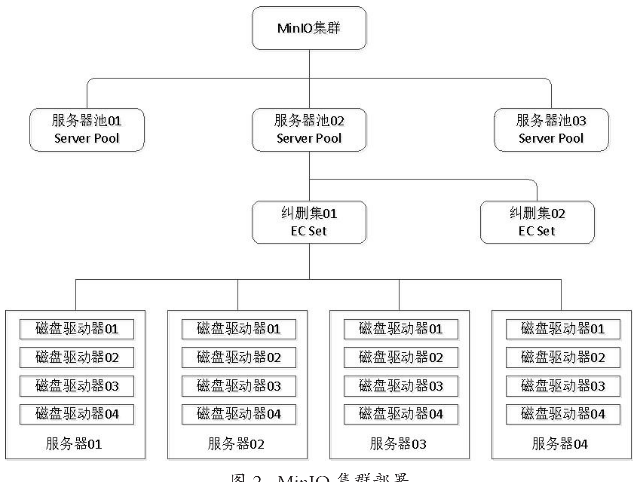


图 2　MinIO 集群部署

除网络外，MinIO 部署设计主要考虑服务器和磁盘数量。由于 MinIO 的分布式集群模式采用纠删码方案存储对象数据，因此 MinIO 集群单个服务器池节点数量和磁盘数量需要符合纠删码的算法特点，各服务器池节点数量和磁盘数量可以不同。按照 MinIO 官网推荐的部署拓扑，一个服务器池内，应具有偶数个节点和磁盘驱动器，两者数量最大公约数应为 16，超过 8 个磁盘驱动器时校验块默认为 4。本文采用 10GbE 网络，8 个节点共 128 个 2TB SAS 磁盘驱动器，EC：4 校验方案，裸容量为 512TB，可用容量为 384TB。

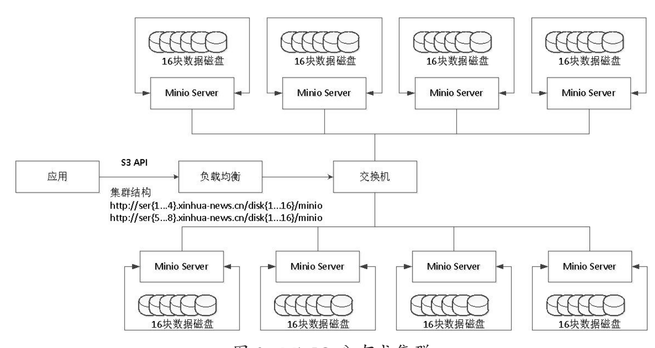


图 3　MinIO 分布式集群

3. 测试与应用

3.1　功能性能测试

使用 COSBench 测试工具分别对 100KB、5MB、100MB 大小的文件进行测试，功能正常，性能测试如下：

对象大小	总容量	指标	读取	写入
100KB（多桶）	100GB	吞吐量（op/s）	229.64	57.36
		带宽（MB/s）	22.96	5.74
5M（多桶）	1000GB	吞吐量（op/s）	19.94	5.13
		带宽（MB/s）	99.68	25.65
100MB（多桶）	500GB	吞吐量（op/s）	1.09	0.22
		带宽（MB/s）	109.14	21.75

图 4　MinIO 测试结果

根据以上测试结果，分布式对象存储读性能较好，除 100KB 小文件外，能够跑满 1Gb 带宽，写性能为读性能的 1/5 到 1/4。

3.2　采编融媒体平台应用

采编业务环境下会产生大量的图片、音频、视频、文档等多媒体文件，多媒体文件的存储、访问及管理是采编业务核心系统的支撑功能。采编业务核心系统中的文件服务是对所有文件资源的管理和存储的关键服务，主要用于统一管理、存储各子系统产出的文件，便于在各子系统之间流转，同时解决由于各子系统独立存储导致的数据冗余问题。文件服务采用 Spring Cloud 架构，分为 4 个独立子服务，分别为静态文件代理服务、文件管理服务、图片处理服务、视频转码服务，

每个子服务都涉及对文件及其元数据的读写。

为此，在采编业务核心系统的文件服务基础上，对以上服务中涉及文件读写的相关接口进行了改造，由最初通过 NAS 的存储路径方式改造成对象存储的 http 接口方式进行访问，新开发了 MinIO 文件上传和文件注册接口，实现了通过 S3 协议的对象存储域名完成文件注册功能。[6] 并模拟获取文件元数据的流程，包括但不限于图片文件的 EXIF 信息、音视频文件的码率时长等信息。

（1）整合 MinIO 添加相关依赖

```
<!-- https://mvnrepository.com/artifact/io.minio/minio -->    <dependency>
      <groupId>io.minio</groupId>
      <artifactId>minio</artifactId>
      <version>8.2.1</version>
</dependency>
```

（2）整合 MinIO 增加 MinIO 配置

```
minio:
  endpoint: 172.21.244.37
  port: 9000
  accessKey: test
  secretKey: DoLTfkduigwklsdfoq836TzcH+Rp4aH5fN1oKuoO6CYT
  secure: false
  bucketName: test-gw
```

其中，endpoint 是 minio api 的 ip，port 是 minio api 的端口号，accessKey 和 secretKey 是 minio 的账号密码，bucketName 表示桶名，可以理解为一个文件夹名称。

（3）定义 MiniO

```
@Component
@ConfigurationProperties ( prefix = "minio" )
public class MinioConfig {
    /*** endpoint 是一个 url、域名、IPv4 或者 IPv6 的地址 */
    private String endpoint;
    private Integer port; /*** 端口号 */
    private String accessKey; // 用户 ID，用于唯一标识你的账户
    private String secretKey; // 私钥
    private Boolean secure; //true 则是 https，false 为 http，默认值 true
    private String bucketName; // 默认存储桶
    @Bean
    public MinioClient minioClient ( ) {
        MinioClient minioClient = MinioClien.builder( )
    .endpoint ( endpoint, port, secure )
    .credentials ( accessKey, secretKey ) .build ( ) ;
        return minioClient;
    }
```

文件上传

```
public void upload ( String objectName, InputStream inputStream ) {
    try {
        long size = inputStream.available ( ) ;
        PutObjectArgs putObjectArgs = PutObjectArgs.builder ( )
                .bucket( minioConfig.getBucketName( ) )
                .object ( objectName ) .stream ( inputStream, size, -1 )
                .build ( ) ;
        minioClient.putObject ( putObjectArgs ) ;
        inputStream.close ( ) ;
    } catch ( IOException e ) {
        e.printStackTrace ( ) ;
    } catch ( Exception e ) {
        e.printStackTrace ( ) ;
    }
}
```

4. 优化实践

根据采编业务核心系统文件服务架构和对象存储的特性和应用，本文验证了对象存储应用于采编业务的可能性，以及反推采编架构层面的优化改进。

在存储层面，使用对象存储替代现有的文件存储（NAS）试验是可行的，但对使用对象存储的其他应用或者服务将会提出新的要求。例如，系统文件服务将由挂载 NAS 存储改为使用 URL 请求访问对象存储资源，每次 I/O 操作将转变为一次请求，尤其是需要全文件获取的操作，如计算 MD5 等。[7] 系统文件服务需要着力考虑 I/O 操作对逻辑流程上带来的影响，需要根据具体情况优化现有流程，以尽可能减少由于对象存储的使用带来的 I/O 压力。

同时，也可以考虑在存储层做一个轻量级的文件元数据管理服务，将文件资源的固有属性信息，如 md5、后缀、类型、图片 EXIF 信息、音视频码率、文件引用次数等，做对应管理。这将极大地减少文件服务和其他服务或系统的 I/O 请求。这也对相关系统间

（下转第158页）